Cost Functions Definable by Min/Max Automata*

Thomas Colcombet[1], Denis Kuperberg[2], Amaldev Manuel[3], and Szymon Toruńczyk[†4]

1 **CNRS & LIAFA, Université Paris Diderot, Paris 7, France**
2 **IRIT/ONERA, Toulouse, France**
3 **MIMUW, University of Warsaw, Poland**
4 **MIMUW, University of Warsaw, Poland**

Abstract

Regular cost functions form a quantitative extension of regular languages that share the array of characterisations the latter possess. In this theory, functions are treated only up to preservation of boundedness on all subsets of the domain. In this work, we subject the well known *distance automata* (also called *min-automata*), and their dual *max-automata* to this framework, and obtain a number of effective characterisations in terms of logic, expressions and algebra.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases distance automata, B-automata, regular cost functions, stabilisation monoids, decidability, min-automata, max-automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.29

1 Introduction

Regular languages enjoy multiple equivalent characterisations, in terms of regular expressions, automata, monoids, monadic second order (MSO) logic, etc. One of them is purely algebraic: a language $L \subseteq A^*$ is regular if and only if its two-sided Myhill-Nerode congruence on A^* has finite index. These characterisations have been refined further for many subclasses of regular languages. The archetypical example is the Schützenberger-McNaughton-Papert theorem. which equates the class of star-free languages (i.e. expressible by star-free regular expressions with complementation), the class of first-order definable languages, the class of languages accepted by *counter-free* automata, and the class of languages definable by aperiodic monoids (i.e. those that satisfy the equation $x^{n+1} = x^n$ for sufficiently large n). This gives an algebraic and effective characterisation of the class of star-free languages.

The theory of regular languages has been extended in many directions – to infinite words, trees, infinite trees, graphs, linear orders, traces, pictures, data words, nested words, timed words etc. With some effort, some of the above characterisations can be transferred, by finding the right notion of a "regular" language, and by finding algebraic and logical characterisations of certain subclasses of languages among the class of all the "regular" languages.

Whereas languages are qualitative objects, in this paper, we study characterisations of classes of quantitative objects. One of the classes that we study are cost functions defined by *distance automata*. A distance automaton $\mathcal{A}$ is like a nondeterministic finite automaton, where each transition additionally carries a *weight*, i.e., a natural number. The weight of

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 259454.
† Author supported by the National Science Center (decision DEC-2012/07/D/ST6/02443).

33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016).
Editors: Nicolas Ollinger and Heribert Vollmer; Article No. 29; pp. 29:1–29:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a run is the sum of the weights of the transitions in the run. The distance automaton $\mathcal{A}$ then associates to each input word w a value $[\![\mathcal{A}]\!](w) \in \mathbb{N} \cup \{\infty\}$, defined as the minimal weight of an accepting run over w, and ∞ if there is no accepting run. The central decision problem is the *limitedness problem* – does the function $[\![A]\!]$ have a finite range?

Distance automata were introduced by Hashiguchi in his solution of the star height problem, which he reduced to the limitedness problem for distance automata via a strenuous reduction. As distance automata try to minimize the value of a run, we call them *min-automata* in this paper. *Max-automata* are the dual model, for which the value of the word is the maximal weight of an accepted run. Min-automata and max-automata have appeared in various contexts (see related work below for more on this) under various names.

Distance automata were extended in subtly distinct ways to nested distance-desert automata [12], R-automata [1], B-automata [4, 6] by allowing several counters instead of just one, and allowing these counters to be reset. The limitedness problem is decidable for all these classes. In terms of cost functions all these extended models are equivalent.

Colcombet [6, 8] discovered that functions defined by B-automata enjoy a very rich theory of *regular cost functions*, extending the theory of regular languages. A *cost function* is an equivalence class of functions, where two functions $f, g : A^* \to \mathbb{N} \cup \{\infty\}$ are equivalent if they are bounded over precisely the same subsets of A^*. A *regular cost function* is the equivalence class of a function computed by a B-automaton. Colcombet gave equivalent characterisations of regular cost functions in terms of a quantitative extension of MSO, an extension of monoids called *stabilisation monoids*, Later, analogous characterisations were described, in terms of a quantitative extension of regular expressions, in terms of logics and regular expressions manipulating profinite words (a completion of the set of finite words in a certain metric) [17], and in terms of a finite index property [17, 14]. In [15] a characterisation in terms of a quantitative extension of FO on the Σ-tree is given.

Contributions

In this paper, we propose a Schützenberger-McNaughton-Papert style characterisation of the subclass of the class of regular cost functions, defined by distance automata – in terms of logic, regular expressions and algebra, i.e., by conditions satisfied by the syntactic stabilisation monoid. The last characterisation provides a machine-independent, purely algebraic description of the cost functions defined by distance automata – or min-automata. We also provide similar characterisations for the dual class of max-automata. Although their definition is simply obtained by replacing min by max, the statements and their proofs are quite different for both classes. Our characterisations are effective, i.e., given a B-automaton, it is decidable whether the cost function it defines is recognisable by a min- or max-automaton. The detailed proofs can be found in the long versions, on the webpage of authors.

Related Work

Characterising special classes of regular cost functions in various formalisms has been done in [11, 14]. In [11], the class of temporal cost functions was defined and studied. These cost functions are only allowed to measure *consecutive* events, for instance the function counting the number of occurrences of a letter in an input word is not temporal. Equivalent characterisations of this class were given in terms of cost automata, regular languages, stabilisation monoids. Additionally, in [7], an equivalent fragment of cost MSO was given. In [14], the class of aperiodic cost functions was considered, as a generalisation of star-free languages. It was shown that this class of function can be equivalently characterised by

definability via cost linear temporal logic, cost first order logic, or group-trivial stabilisation monoids, generalising the Schützenberger-McNaughton-Papert theorem to cost functions.

In the papers [5] and [3], min- and max-automata were defined in a different way, as *deterministic* automata with many counters, over which any sequence of instructions could be performed in a single transition, where each instruction is either of the form $c := c + 1$ or $c := \max(d, e)$ (in the case of max-automata) or $c := \min(d, e)$ (in the case of min-automata), where c, d, e are counters. As these models were studied in relationship with logics over infinite words, rather than evaluating a finite word to a number, they were used as acceptors of infinite words. However, their finitary counterparts are equivalent to the models studied in this paper, up to cost function equivalence (see Proposition 2.4). Note that nondeterminism is exchanged for multiple counters with aggregation (min or max).

In the paper [2], Cost Register Automata are studied, and are parametrised by a set of operations. Those too are deterministic automata with many registers (i.e. counters). For the set of operations denoted $(\min, +c)$, one obtains a model equivalent to the min-automata of [5], and for the set of operations denoted $(\max, +c)$, one obtains a model equivalent to the max-automata [3].

Min-automata can be equivalently described as nondeterministic weighted automata over the semiring $(\mathbb{N} \cup \{\infty\}, \min, +)$, where min plays the role of addition and $+$ of multiplication. Similarly, max automata can be equivalently described as nondeterministic weighted automata over the semiring $(\mathbb{N} \cup \{\infty, \bot\}, \max, +)$, where max plays the role of addition and $+$ of multiplication (and $\bot$ is neutral with respect to max and absorbing with respect to $+$). Using this formalism, decidability results about precision of approximation of functions computed by min-automata are shown in [9]. Similar results on max-automata are presented in [10], together with an application to evaluation of time complexity of programs.

2 Preliminaries

In this section, we recall various models of cost automata, and the theory of regular cost functions. For more details, the reader should confer [6, 8, 11]. We write $\mathbb{N}_\infty$ for the set $\mathbb{N} \cup \{\infty\}$. We follow the convention that $\inf \emptyset = \min \emptyset = \infty$ and $\sup \emptyset = \max \emptyset = 0$.

2.1 Automata

We recall the notions of B- and S-automata, and relate them to min- and max-automata.

B-automata and S-automata. *B- and S-automata* are nondeterministic automata over finite words, which are moreover equipped with a finite set of *counters*. Each counter admits three basic operations: incrementation by one, denoted `i`, reset to zero, denoted `r`, and the idle operation, denoted ε. Initially, all counters are set to 0, and during the run each transition of the automaton performs one operation on each counter separately (formally, a transition is a tuple (p, a, o, q), where p is its source state, q is its target state, a is the label and $o = (o_c)_c$ is a vector of operations, one per each counter c). Additionally, we allow counters to be reset after the run terminates in an accepting state, depending on the state. If in a run ρ some transition resets a counter currently storing a value n, then we say that n is a *reset value* for the considered run ρ.

We now define the value of a word under a B-automaton; the defintion for S-automata is dual and will be given later. Let $\mathcal{B}$ be a B-automaton and w be an input word. For a run ρ

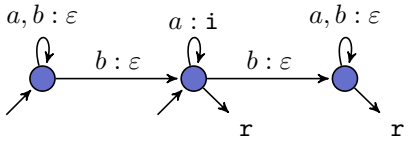

Figure 1 A B-automaton, which is also a min-automaton. Edges with no source state mark initial states, and edges without a target state mark accepting states and may reset the counter.

over the word w, define the *cost* of ρ as the maximum of the set of its reset values:

$$\text{cost}(\rho) = \max\{n \in \mathbb{N} : n \text{ is a reset value for } \rho\}.$$

Recall that the maximum of the empty set is equal to 0. Finally, define $[\![\mathcal{B}]\!](w)$ as the minimal value of an accepting (initial-to-accepting state) run over w:

$$[\![\mathcal{B}]\!](w) \;=\; \min\{\text{cost}(\rho) \mid \rho \text{ is an accepting run of } \mathcal{B} \text{ over } w\}.$$

Note that this value can be infinite, if there is no accepting run of $\mathcal{B}$ over w. In particular, if the set of counters is empty, then $[\![\mathcal{B}]\!](w) = 0$ if $\mathcal{B}$ has an accepting run over w, otherwise $[\![\mathcal{B}]\!](w) = \infty$. In this way, B-automata generalize finite automata.

If $\mathcal{A}$ is an S-automaton, the definitions are obtained by swapping min with max. In particular, the cost of the run is the minimum of the set of its reset values (recall that $\min \emptyset = \infty$) and $[\![\mathcal{A}]\!](w)$ is the maximal value of an accepting run over w. If $\mathcal{A}$ has no counters, then $[\![\mathcal{A}]\!](w) = \infty$ if $\mathcal{A}$ has an accepting run over w, otherwise $[\![\mathcal{A}]\!](w) = 0$.

For a B- or S-automaton $\mathcal{A}$, we call $[\![\mathcal{A}]\!]$ the function *computed* by $\mathcal{A}$.

▶ **Example 2.1.** We construct a B-automaton that computes the smallest length of a block of consecutive a's in an input word consisting of a's and b's. In other words, for an input word w of the form $a^{n_1}ba^{n_2}\dots ba^{n_k}$, the computed value is $f_{\min}(w) = \min_{j=1}^{k} n_j$. The automaton has one counter, and is depicted in Figure 1.

▶ **Example 2.2.** Another example of a function computed by a B-automaton is the following. For a given word w over the alphabet $\{a, b, c\}$ of the form $w_1cw_2\dots cw_k$, where the words $w_1, \dots, w_k$ are over the alphabet $\{a, b\}$, define $f(w) = \max_{j=1}^{k} f_{\min}(w_j)$. The function f is computed by a B-automaton obtained from the automaton in Figure 1 by adding a c-labeled, resetting transition from every accepting state to every initial state.

▶ **Example 2.3.** The automaton in Figure 1 can be interpreted as an S-automaton which, for an input word w of the form $a^{n_1}ba^{n_2}\dots ba^{n_k}$, computes the value $f_{\max}(w) = \max_{j=1}^{k} n_j$.

Min-automata and max-automata. A *min-automaton* is a one-counter B-automaton $\mathcal{B}$, with only two operations allowed: `i` (increment) and ε (do nothing). In particular, resets are not allowed during the run. However, every counter is reset at the end of the run. Therefore the last counter value is a reset value. In other words, a min-automaton is a nondeterministic finite automaton in which every edge carries one of the two operations `i` or ε which manipulate the only counter. The cost of a run is the last value attained by the counter, and $[\![\mathcal{B}]\!](w)$ is the minimum of the costs of all accepting runs. This corresponds exactly to the definition of a *distance automaton* given in the introduction, with `i` corresponding to 1 and ε corresponding to 0 in the distance automaton. The automaton from Example 2.1 is a min-automaton.

Dually, a *max-automaton* $\mathcal{A}$ is a one-counter S-automaton, with only the two operations $\mathtt{i}$ and ε allowed, and where the automaton may, depending of the last state assumed, reset or not the counter at the end of the run. Therefore, the cost of a run is again the last value attained by the counter if it is reset, and $+\infty$ if it is not reset. The value of a word $[\![\mathcal{A}]\!](w)$ is the maximum of the costs of all accepting runs. Example 2.3 gives an example of a max-automaton.

As mentioned in the related work in the introduction, min/max-automata are related to other notions from the literature. We establish this connection in the proposition below, and later on in this paper, we will only talk about min- and max-automata.

A *weighted automaton* over a semiring $\mathcal{S}$ specifies a $n \times n$ matrix $h(a)$ over $\mathcal{S}$ for each letter a in the input alphabet, and two vectors I, F of length n over $\mathcal{S}$. The *value* associated to a word $a_1 \dots a_l$ over the input alphabet is the product of the matrices $I^T \cdot h(a_1) \cdots h(a_l) \cdot F$, which is a 1×1 matrix, identified with an element of $\mathcal{S}$. In the proposition below, a value $\bot$ returned by a weighted automaton is interpreted as 0.

▶ **Proposition 2.4** ([2, 5]). *Min-automata are equivalent to distance automata, to nondeterministic weighted automata over the semiring* $(\mathbb{N}_\infty, \min, +)$*, and to deterministic automata with many registers storing elements of* $\mathbb{N}_\infty$*, allowing the binary* min *operation and unary incrementation operation.*

Dually, max-automata are equivalent to nondeterministic weighted automata over the semiring $(\mathbb{N}_\infty \cup \{\bot\}, \max, +)$*, and to deterministic automata with many registers storing elements of* $\mathbb{N}_\infty$*, allowing the binary* max *operation and unary incrementation operation.*

The goal of this paper is to find effective algebraic characterisations of functions computable by min- and max-automata amongst all functions computable by B- and S-automata. These characterisations are up to equivalence of cost functions.

2.2 Theory of Regular Cost Functions

Throughout this paper, fix a finite input alphabet Σ. Given two functions $f, g : \Sigma^* \to \mathbb{N}_\infty$, we write $f \approx g$ if for all $X \subseteq \Sigma^*$, if g is bounded over X (meaning $\sup g|_X < \infty$), then f is bounded over X, and vice-versa. A *cost function* over the alphabet Σ is an equivalence class of $\approx$. Let $[f]$ denote the equivalence class of $f : \Sigma^* \to \mathbb{N}_\infty$. We will often identify a cost function with any of its representatives and say that f is a cost function, implicitly talking about $[f]$.

Regular cost functions. A cost function is *regular* if it is the cost function of the function computed by some B-automaton. For example, the (equivalence classes of the) functions described in Examples 2.1 and 2.2 are regular cost functions. It turns out [6] that B- and S-automata define equal classes of cost functions (see Theorem 2.15 below). Not every cost function is regular – indeed, there are uncountably many cost functions.

The reason we prefer to study cost functions computed by B-automata rather than the functions themselves is due to the following results, and to the fact that information about boundedness properties suffice in the contexts we will be interested in.

▶ **Theorem 2.5** (Krob [13]). *Given two min-automata* $\mathcal{A}$ *and* $\mathcal{B}$*, it is undecidable whether the functions* $[\![\mathcal{A}]\!]$ *and* $[\![\mathcal{B}]\!]$ *are equal.*

▶ **Theorem 2.6** (Colcombet [6]). *Given two B- or S-automata* $\mathcal{A}$ *and* $\mathcal{B}$*, it is decidable whether the functions* $[\![\mathcal{A}]\!]$ *and* $[\![\mathcal{B}]\!]$ *define the same cost function.*

If we take $\mathcal{B}$ in the theorem above to be such that $[\![\mathcal{B}]\!](w) = 0$ for all words w, we see that in particular, it is decidable whether the function $[\![\mathcal{A}]\!]$ is bounded over all words. The limitedness problem for B-automata easily reduces to this problem.

Cost regular expressions. Cost regular expressions are weighted extensions of classical regular expressions. They come in two forms, B-expressions and their dual S-expressions. A *B-expression* is given by the grammar,

$$E ::= a \in \Sigma \mid \emptyset \mid E \cdot E \mid E + E \mid E^{\leq n} \mid E^*,$$

where n is a variable (there is only one variable available). Note that by substituting $k \in \mathbb{N}$ for n in a cost regular expression E, denoted by $E[k \to n]$, one obtains a regular expression of finite words. Given a B-expression E the cost function computed by E is defined as:

$$[\![E]\!](u) = \inf\{k \mid u \in E[k \to n]\}.$$

Similarly one defines S-expressions, with the difference that we are allowed to use $> n$ instead of $\leq n$, and at the end we take sup instead of inf.

Both kinds of expressions define exactly all regular cost functions. Indeed, it is not difficult to convert between B-expressions and B-automata (and between S-expressions and S-automata), similarly to the conversions between regular expressions and finite automata.

▶ **Example 2.7.** The cost function $f_{\max}$ is defined by the B-expression $(a^{\leq n}b)^*$ and the S-expression $(a^*b)^*a^{>n}(ba^*)^*$. Dually, the cost function $f_{\min}$ is defined by the B-expression $(a^*b)^*a^{\leq n}(ba^*)^*$ and the S-expression $(a^{>n}b)^*a^{>n}$.

Cost monadic second order logic. We recall the basics of monadic second order logic (abbreviated as MSO) over words. The formulas use first order variables $x, y, z \ldots$ that range over positions of the word and second order variables $X, Y, Z \ldots$ that range over sets of positions of the word. MSO formulas are build using the atomic predicates $x \leq y$, $x \in X$, $a(x)$ (denoting that the label at position x is a, where $a \in \Sigma$), the connectives $\neg\varphi$, $\varphi \vee \psi$, $\varphi \to \psi$ and quantifiers $\exists x.\varphi$, $\forall x.\varphi$ (ranging over single elements) and $\exists X.\varphi$,$\forall X.\varphi$ (ranging over sets of elements). *Cost monadic second order logic* (cost-MSO) extends the logic by allowing formulas of the form $|X| \leq n$, where $|X|$ denotes the size of the set X and n is a fixed variable ranging over the natural numbers, with the restriction that they occur only positively (under even number of negations). Given a cost-MSO formula $\varphi(n)$ the cost function defined by $\varphi(n)$ is

$$[\![\varphi]\!](u) = \inf\{n \mid u, n \models \varphi\}.$$

▶ **Example 2.8.** The cost function $f_{\min}$ is expressed by $\exists X.block_a(X) \wedge (|X| \leq n)$, where $ablock_a(X)$ is the first-order formula expressing that X is a maximal block of consecutive a's. Dually, $f_{\max}$ is expressed by the formula $\forall X.block_a(X) \to (|X| \leq n)$.

Stabilisation monoids. Recall that if $\mathbf{M}$ is a finite monoid, $h : \Sigma \to \mathbf{M}$ is a mapping, and F is a subset of $\mathbf{M}$, then the triple $(\mathbf{M}, h, F)$ defines a regular language $L = \hat{h}^{-1}(F)$, where $\hat{h} : \Sigma^* \to \mathbf{M}$ is the unique homomorphism extending h. Conversely, any regular language L is induced by some triple $(\mathbf{M}, h, F)$ of this form. In this section we recall how this correspondence lifts to regular cost functions, by replacing finite monoids to stabilisation monoids. Let $E(\mathbf{M}) = \{e \in M | ee = e\}$ denote the set of idempotents of the monoid $\mathbf{M}$.

A *stabilisation monoid* $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ is a finite monoid equipped with a partial order $\leq$ and an operation $\sharp : E(\mathbf{M}) \to E(\mathbf{M})$ (called stabilisation), satisfying the following axioms:

$$a \cdot x \leq b \cdot y \quad \text{for} \quad a \leq b, x \leq y \tag{1}$$
$$e^\sharp \leq f^\sharp \quad \text{for} \quad e \leq f, e, f \in E(\mathbf{M}) \tag{2}$$
$$(a \cdot b)^\sharp = a \cdot (b \cdot a)^\sharp \cdot b \quad \text{for} \quad a, b \in \mathbf{M} \text{ such that } a \cdot b, b \cdot a \in E(\mathbf{M}) \tag{3}$$
$$e^\sharp \leq e \quad \text{for} \quad e \in E(\mathbf{M}) \tag{4}$$
$$(e^\sharp)^\sharp = e^\sharp \quad \text{for} \quad e \in E(\mathbf{M}) \tag{5}$$

▶ **Example 2.9.** Any finite monoid can be seen as a stabilisation monoid, in which $e^\sharp = e$ for every idempotent, and where the order is trivial, i.e., $x \leq y$ iff $x = y$.

▶ **Example 2.10.** Every min-automaton $\mathcal{A}$ defines a finite transition stabilisation monoid, defined as follows. Let $\mathcal{T}$ denote the *tropical semiring* or the $(min, +)$-semiring, with domain $\mathbb{N}_\infty$ and min playing the role of addition, and $+$ of multiplication. We equip $\mathbb{N}_\infty$ with the usual topology, in which ∞ is the limit of every strictly increasing sequence.

First we define an infinite monoid, parametrised by a finite set of states Q. Let $\mathbf{M}_Q$ denote the set of $Q \times Q$ matrices with entries from $\mathcal{T}$. Matrices can be multiplied using the semiring operations of $\mathcal{T}$, and the set of matrices $\mathbf{M}_Q$ inherits the product topology from $\mathcal{T}$, i.e., a sequence $(M_n)_{n=1}^\infty$ of matrices is convergent if $M_n[p, q]$ is convergent in $\mathbb{N}_\infty$ for each $p, q \in Q$. Matrix multiplication is continuous, and moreover, one can show that for every matrix $M \in \mathbf{M}_Q$, when $n \to \infty$, the sequence $M^{n!}$ is convergent to a matrix denoted $M^\sharp$; moreover, the mapping $M \mapsto M^\sharp$ is continuous.

An automaton $\mathcal{A}$ with state space Q defines a mapping $h : \Sigma^* \to M$ which assigns to a word $w \in \Sigma$ the matrix $h(w)$ such that for two states p, q of $\mathcal{A}$, the value $h(w)[p, q]$ is the minimal cost of a run of $\mathcal{A}$ over w which starts in state p, ending in state q. The mapping h is a monoid homomorphism.

Define an equivalence relation $\sim_1$ on $\mathbf{M}_Q$ so that two matrices are equivalent iff they yield the same result when each finite, positive entry is replaced by 1. Define $\mathbf{M}_Q^1$ to be the set of $\sim_1$-equivalence classes. We identify an element of $\mathbf{M}_Q^1$ with its unique representative which is a matrix with entries in $\{0, 1, \infty\}$. It turns out [17] that the equivalence $\sim_1$ preserves multiplication and the operation $\sharp$. It follows that $\mathbf{M}_Q^1$ inherits the structure of a monoid, and also an operation $\sharp$; we restrict this operation only to idempotents (for a non-idempotent M, we can still recover $M^\sharp$ as $E^\sharp$, where E is the idempotent power of M).

One way to compute a product of two matrices $M, N \in \mathbf{M}_Q^1$ is to take a min-automaton $\mathcal{A}$ with states Q and with $h(a) = M$ and $h(b) = N$; then $M \cdot N$ is obtained by substituting 1 for every finite positive number in $h(ab)$. Similarly, if $M \in \mathbf{M}_Q^1$ is idempotent, then $M^\sharp$ is obtained by taking an automaton $\mathcal{A}$ with $h(a) = M$ and writing $M^\sharp[p, q] = 0$ if for arbitrarily large n, $h(a^n)[p, q] = 0$, otherwise $M^\sharp[p, q] = 1$ if for arbitrarily large n $h(a^n)[p, q]$ remains bounded, and finally, $M^\sharp[p, q] = \infty$ if $h(a^n)[p, q]$ converges to ∞ when $n \to \infty$. The computations can be performed purely mechanically (see Appendix). For example, for the automaton from Example 2.1, we compute $h(a) \cdot h(b)$ and $h(a)^\sharp$ in $\mathbf{M}_Q^1$:

$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & 1 & \infty \\ \infty & \infty & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & \infty & \infty \\ 0 & \infty & \infty \\ \infty & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \infty & \infty \\ 1 & \infty & \infty \\ \infty & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & \infty & \infty \\ \infty & 1 & \infty \\ \infty & \infty & 0 \end{bmatrix}^\sharp = \begin{bmatrix} 0 & \infty & \infty \\ \infty & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

For $M, N \in \mathbf{M}_Q^1$, write $M \preceq N$ if there is a sequence of representatives of N which converges to a representative of M. In other words, M can be obtained from N by replacing some 1's

by ∞'s. (The order $\preceq$ is the specialisation order associated to the quotient topology on $\mathbf{M}_Q^1$ inherited from $\mathbf{M}_Q$.)

It can be shown [17] that $\langle \mathbf{M}_Q^1, \cdot, \preceq, \sharp \rangle$ is a stabilisation monoid. The axiom (1) is a remnant of continuity of multiplication in $\mathbf{M}_Q$, (2) – of continuity of the operation $M \mapsto M^\sharp$, (3) – of associativity, (4) – of the fact that $M^\sharp$ is the limit of $M^{n!}$, and (5) – of an analogous property which holds in $\mathbf{M}_Q$.

Let $h^1 : \Sigma \to \mathbf{M}_Q^1$ be defined so that for a letter $a \in \Sigma$, the matrix $h^1(a)$ is equal to $h(a)$ (this is a matrix with entries in $\{0, 1, \infty\}$). The mapping h^1 encodes the transition structure of the automaton $\mathcal{A}$. Let I be the set of matrices $M \in \mathbf{M}_Q^1$ such that $M[p, q] = \infty$ for all initial states p and accepting states q of $\mathcal{A}$. The set I encodes the acceptance condition of $\mathcal{A}$. The cost function $[\![\mathcal{A}]\!]$ defined by $\mathcal{A}$ can be recovered from the triple $(\mathbf{M}_Q^1, h^1, I)$, as we will now describe.

▶ **Definition 2.11** (Computation tree)**.** An *n-computation tree* t is a finite rooted ordered unranked tree in which each node x has an associated *output* in $\mathbf{M}$ and is of one of four types:

Leaf x has no children and has an associated *label* $a \in \Sigma$, and the output of x is $h(a)$;

Binary node x has exactly two children and and the output of x is the product of the output of the first child and the output of the second child;

Idempotent node x has k children with $k \leq n$ and for some idempotent $e \in \mathbf{M}$, the output of each child is equal to e and the output of x is equal to e.

Stabilisation node x has k children with $k > n$ and for some idempotent $e \in \mathbf{M}$, the output of each child is equal to e and the output of x is equal to $e^\sharp$.

The *input* of the tree t is the word formed by the labels of the leaves of the tree, read left to right. The *output* of the tree t is the output of the root, and the neutral element of $\mathbf{M}$ if t is the empty tree.

An *ideal* in a stabilisation monoid $\mathbf{M}$ is a subset I which is downward-closed, i.e., $x \leq y$ and $y \in I$ imply $x \in I$. Let $h : \Sigma \to M$ be a mapping from a finite alphabet to a stabilisation monoid $\mathbf{M}$, and let I be an ideal in $\mathbf{M}$. The triple $(\mathbf{M}, h, I)$ induces a cost function, denoted $[\![\mathbf{M}, h, I]\!]$, and defined as follows. For a fixed height $k \in \mathbb{N}$, let

$$[\![\mathbf{M}, h, I]\!]_k (w) = \inf\{n \mid \text{there is a } n\text{-computation on } w \text{ with output in } \mathbf{M} \setminus I, \text{ height} \leq k\}.$$

It turns out [6] that the cost function $[\![\mathbf{M}, h, I]\!]_k$ does not depend on the choice of $k \geq 3|M|$. We define $[\![\mathbf{M}, h, I]\!]$ as $[\![\mathbf{M}, h, I]\!]_k$ for $k = 3|M|$.

▶ **Example 2.12.** Let $\mathbf{M}_{\max} = \langle \{1, a, b, a^\sharp\}, \cdot, \sharp, \leq \rangle$ be the stabilisation monoid with identity 1, zero $a^\sharp$, such that $ab = ba = b = bb = b^\sharp$, $aa = a$ and $a^\sharp \leq a$. The monoid $\mathbf{M}_{\max}$ defines the function $f_{\max}$ with ideal $\{a^\sharp\}$ and mapping $h(a) = a$ and $h(b) = b$.

▶ **Example 2.13.** Let $\mathbf{M}_{\min} = \langle \{1, a, a^\sharp, b, ba^\sharp, a^\sharp b, 0\}, \cdot, \sharp, \leq \rangle$ be the stabilisation monoid with identity 1, zero 0, product $a^\sharp ba^\sharp = a^\sharp$, $ba^\sharp b = b = ab = ba$, $bb = 0$, stabilisation $(ba^\sharp)^\sharp = ba^\sharp, (a^\sharp b)^\sharp = a^\sharp b$, and order $a^\sharp \leq a, a^\sharp b \leq b, ba^\sharp \leq b \leq 0$. The monoid $\mathbf{M}_{\min}$ defines the function $f_{\min}$ with ideal $\{a^\sharp\}$ and mapping $h(a) = a$ and $h(b) = b$.

▶ **Proposition 2.14.** *For a min-automaton $\mathcal{A}$ with states Q, define $(\mathbf{M}_Q^1, h^1, I)$ as in Example 2.10. Then I is an ideal and the cost functions $[\![\mathbf{M}_Q^1, h^1, I]\!]$ and $[\![\mathcal{A}]\!]$ are equal.*

Example 2.10 and Proposition 2.14 are a special case of a more general construction for B- and S-automata [17].

Closure properties. Regular cost functions have several closure properties, described below. The fundamental cost function is the function count : $\{a,b\} \to \mathbb{N}_\infty$, defined by $\text{count}(u) = |u|_a$, the number of occurrences of letter a. A regular language $L \subseteq \Sigma^*$ is viewed as a (regular) cost function mapping a word w to 0 if $w \in L$ and to ∞ if $w \notin L$. Note that since regular languages are closed under complements, exchanging 0 with ∞ in this definition would give the same class of cost functions.

Let $\mathcal{C}$ be a class of cost functions, possibly over different input alphabets. We define several closure properties for the class $\mathcal{C}$:

composition with morphisms if the cost function $f : \Sigma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$ and $\alpha : \Gamma^* \to \Sigma^*$ is a morphism, then the cost function $\alpha \circ f : \Gamma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$;

min if for any two cost functions $f, g : \Sigma^* \to \mathbb{N}_\infty$ in $\mathcal{C}$, the function $w \mapsto \min(f(w), g(w))$ also belongs to $\mathcal{C}$;

max defined dually, with max instead of min;

min with regular languages if for any cost function $f : \Sigma^* \to \mathbb{N}_\infty$ and regular language g viewed as a cost function (see above) the function $w \mapsto \min(f(w), g(w))$ also belongs to $\mathcal{C}$;

sup-projections if the cost function $f : \Sigma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$ and $\alpha : \Sigma^* \to \Gamma^*$ is a morphism, then the cost function $v \mapsto \sup\{f(w) : w \in \alpha^{-1}(v)\}$ is in $\mathcal{C}$;

inf-projections defined dually, with inf instead of sup.

The main theorem of regular cost functions. The theorem below shows that all the introduced notions give rise to the same class of cost functions, namely regular cost functions.

▶ **Theorem 2.15** (Colcombet [7]). *The following formalisms are effectively equivalent as recognisers of regular cost functions: B-automata, S-automata, B-expressions, S-expressions, cost MSO formulas, and stabilisation monoids. Moreover, the class of regular cost functions is the smallest class of cost functions which is closed under min, max, inf-projections, sup-projections, contains the function count, and all regular languages.*

We may now specify the goal of this paper more precisely. Among all regular cost functions, we characterise those which are of the form $[\![\mathcal{A}]\!]$ for a min- or max-automaton $[\![\mathcal{A}]\!]$. Our characterisations (Theorem 3.2 and Theorem 4.2) are in terms of cost regular expressions, fragments of cost MSO, and stabilisation monoids. Moreover, the characterisations are effective. In algebraic language theory, the usual way of providing effective characterisations is by means of certain algebraic conditions satisfied by the syntactic monoid. For this reason, we need to recall the notion of a syntactic stabilisation monoid.

Syntactic stabilisation monoid. A homomorphism of stabilisation monoids is a mapping $h : \mathbf{M} \to \mathbf{N}$ of stabilisation monoids which is monotone (i.e., $u \leq v \implies h(u) \leq h(v)$), preserves multiplication (i.e., $h(u \cdot v) = h(u) \cdot h(v)$) and stabilisation (i.e., $h(e^\sharp) = h(e)^\sharp$ for every idempotent $e \in \mathbf{M}$).

▶ **Theorem 2.16** ([11]). *Let $f : \Sigma^* \to \mathbb{N}_\infty$ be a regular cost function. There is a unique (up to isomorphism) triple $(\mathbf{M}_f, h_f, I_f)$ recognising f with the following property. For any triple $(\mathbf{M}, h, I)$ recognising f, there is a unique surjective homomorphism $\phi : \mathbf{M} \to \mathbf{M}_f$ such that $h_f = \phi \circ h$ and $I = \phi^{-1}(I_f)$. $\mathbf{M}_f$ is called the syntactic stabilisation monoid of f.*

Moreover, for any triple $(\mathbf{M}, h, I)$ recognising f, the triple $(\mathbf{M}_f, h_f, I_f)$ can be computed in polynomial time from $(\mathbf{M}, h, I)$.

Idempotent power. It is a standard fact that every element a in a finite monoid has a unique idempotent power, denoted a^ω. It can be shown that $a^\omega = a^{n!}$ where n is the size of the monoid. We use the notation $a^{\omega\sharp}$ to denote the element $(a^\omega)^\sharp$.

3 Max-automata

In this section we characterise cost functions computed by max-automata. First we introduce the algebraic condition that characterises this class.

▶ **Definition 3.1** (Max-property). Let $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ be a stabilisation monoid and $I \subseteq M$ be an ideal. The pair M, I has *Max-property* if for every $u, v, x, y, z \in M$,

1. if $xu^{\omega\sharp}yv^{\omega\sharp}z \in I$, then either $xu^{\omega\sharp}yv^{\omega}z \in I$, or $xu^{\omega}yv^{\omega\sharp}z \in I$, and
2. if $x(uy^{\omega\sharp}v)^{\omega\sharp}z \in I$, then either $x(uy^{\omega\sharp}v)^{\omega}z \in I$, or $x(uy^{\omega}v)^{\omega\sharp}z \in I$.

Now we present the main theorem of Section 3.

▶ **Theorem 3.2.** *The following are effectively equivalent for a regular cost function f:*

1. *f is accepted by a max-automaton,*
2. *f is definable by a formula of the form $\psi \wedge \forall X\ (\varphi(X) \to |X| \leq n)$ where ψ, φ are MSO formulas, i.e. they do not contain cost predicates,*
3. *f is in the smallest class (call it MAX) of cost functions that contains the function count and regular languages, and closed under min with regular languages, max, sup-projections, and composition with morphisms,*
4. *f is equivalent to $[\![\mathbf{M}, h, I]\!]$ for some mapping h from Σ to a stabilisation monoid $\mathbf{M}$ with ideal I having Max-property,*
5. *The syntactic stabilisation monoid and ideal of f has Max-property,*
6. *f is definable by an S-regular expression of the form $h + \sum_i e_i$ where h is a regular expression and each e_i is of the form $ef^{>n}g$ where e, f and g are regular expressions.*

▶ **Example 3.3.** The stabilisation monoid $\mathbf{M}_{\max}$ with ideal $\{a^\sharp\}$ recognising $f_{\max}$ has Max-property. But $\mathbf{M}_{\min}$ with ideal $I = \{a^\sharp\}$ recognising $f_{\min}$ violates the Max-property since $a^\sharp ba^\sharp = a^\sharp$ is in I, but neither of $a^\sharp ba = a^\sharp b, aba^\sharp = ba^\sharp$ belongs to I.

Since $\mathbf{M}_{\min}$ is the syntactic monoid of the function $f_{\min}$, the example above implies that $f_{\min}$ is not accepted a max-automata; in particular, max-automata are not closed under inf-projection. Similarly one verifies that cost functions computed by max automata are not closed under min (for instance the functions $u \in \{a, b\}^* \to |u|_a$ and $u \in \{a, b\}^* \to |u|_b$, as well as their max, is in MAX, but not their min).

Proof sketch. We sketch the implications: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 1$.

$1 \to 2$ is by observing that there is a cost MSO formula encoding the runs of a 1-counter S-automata, of the form described in item 2.

To prove $2 \to 3$, it is enough to observe that all the subformulas (including the formula itself) of $\psi \wedge \forall X\ (\varphi \to |X| \leq n)$ (where ψ, φ are cost free) define cost functions in the class MAX. For this, we remark that the cost function count is in MAX, so it is also the case of $[\![|X| \leq n]\!]$, using the usual semantic for formulae with free variables, i.e. enriching the alphabet with a $\{0, 1\}$-component. Moreover MAX is closed under the operation min with regular languages, composition with morphisms (together they imply $[\![\neg\varphi \vee |X| \leq n]\!]$ is in MAX) and sup-projections (hence $[\![\forall X\ (\varphi(X) \to |X| \leq n)]\!]$ is in MAX). Finally closure under max implies that the formula defines a function in MAX.

To show $3 \to 4$, we show that the monoid constructions corresponding to the operations of max, min with a regular language, sup-projection and composition with a morphism preserve the Max-property, and also that the syntactic stabilisation monoids computing the function count as well as regular languages have the Max-property.

$4 \to 5$ follows from Theorem 2.16. Implication $5 \to 6$ is the hardest part of the theorem. By definition, the value of a word w given by the cost function $[\![\mathbf{M}, h, I]\!]$ is the maximum $n \in \mathbb{N}_\infty$ such that there is an n-computation tree of height $3|M|$ with input w and output in the ideal I. A way to attain this value using a cost regular expression is to write an expression that encodes all such computation trees by induction on the tree height; binary nodes translate to concatenation, idempotent nodes stand for Kleene star, and stabilisation nodes translate to the operator $>n$. But as such, this idea results in an expression with multiple occurrences of $>n$ (as there could be many stabilisations in the tree). This difficulty is circumvented by showing that it is sufficient to consider trees with only one stabilisation node. This is achieved by repeated use of the Max-property of the monoid $\mathbf{M}$ and ideal I.

For $6 \to 1$, note that the standard construction from cost regular expressions to S-automata (analogous to the translation from regular expressions to finite state automata) applied to the cost regular expressions of the form described in item 6, gives a max-automaton.

We note that all the transformations are effectively computable. ◀

Given a stabilisation monoid $\mathbf{M}$ and ideal I it is computable in polynomial time whether $\mathbf{M}, I$ has Max-property. Hence by Theorem 2.15 and Theorem 2.16 we obtain,

▶ **Theorem 3.4.** *It is decidable if a regular cost function satisfies a condition of Theorem 3.2.*

4 Min-automata

In this section we characterise cost functions definable by min-automata.

▶ **Definition 4.1** (Min-property)**.** We define the relation $\mathcal{R} \subseteq \mathbf{M} \times \mathbf{M}$ as the smallest reflexive relation satisfying the following implications: (1) if $x \mathcal{R} y$ and $a \mathcal{R} b$, then $(x \cdot a) \mathcal{R} (y \cdot b)$, and (2) if $x \mathcal{R} y$ then $x^\omega \mathcal{R} y^{\omega\sharp}$ and $x^{\omega\sharp} \mathcal{R} y^{\omega\sharp}$. A monoid $\mathbf{M}$ satisfies the *Min-property* if for all elements x, y, if $x^{\omega\sharp} \mathcal{R} y$, then $x^{\omega\sharp} = x^{\omega\sharp} y x^{\omega\sharp}$.

▶ **Theorem 4.2.** *The following are effectively equivalent for a regular cost function f:*

1. *f is accepted by a min-automaton,.*
2. *f is definable by a formula of the form $\exists X\ (\varphi(X) \wedge |X| \leq n)$ where φ does not contain any cost predicates,*
3. *f belongs to the smallest class of cost functions containing count and regular languages that is closed under min, max and inf-projections,*
4. *f is recognised by a stabilisation monoid $\mathbf{M}$ with Min-property,*
5. *The syntactic stabilisation monoid of f has Min-property,*
6. *f is accepted by a B-regular expression E that is generated by the grammar*

$$E := F \mid E + E \mid E \cdot E \mid E^{\leq n} \qquad\qquad F := a \mid F + F \mid F \cdot F \mid F^*$$

   i.e. any subexpression of E of the form F^* is a regular expression (without $X^{\leq n}$),
7. *f is accepted by a B-automaton without reset.*

▶ **Example 4.3.** The monoid $\mathbf{M}_{\max}$ of $f_{\max}$ violates the Min-property since $b = b^\sharp \mathcal{R} a^\sharp$ (to see this, observe $a \mathcal{R} a^\sharp, b \mathcal{R} b$ and hence $ab = b \mathcal{R} a^\sharp = a^\sharp b$), but $b = b^\sharp \neq b^\sharp a^\sharp b^\sharp = b a^\sharp b = a^\sharp$. On the contrary, it can be verified that the monoid $\mathbf{M}_{\min}$ has Min-property.

Whereas max automata are not closed under min, min automata are closed under max. The class MIN falls only short of sup-projections, comparing to all regular cost functions.

The Min-property can be expressed in terms of identities, as follows. Consider the set T of terms involving variables from an infinite set of variables, a binary multiplication operation and unary operations ω and $\omega\sharp$. Let $\mathcal{R}$ be the smallest binary relation on T that is reflexive and satisfies the implications 1 and 2 from Definition 4.1. Then, Min-property is expressed as the family of identities $x^{\omega\sharp} = x^{\omega\sharp}yx^{\omega\sharp}$, indexed by pairs of terms x, y such that $x^{\omega\sharp}\mathcal{R}y$.

Proof sketch. We sketch the implications: $1 \to 2 \to 3 \to 1 \to 4 \to 5 \to 1 \to 6 \to 7 \to 1$.

$1 \to 2$, $2 \to 3$, $6 \to 7$ are analogous to the corresponding cases in Theorem 3.2, and $3 \to 1$ is demonstrated by verifying that all functions and operations in item 3 can be carried out in the framework of min-automata.

$1 \to 4$ proceeds by studying the transition monoid of a min-automaton, described in Example 2.10, and checking that the equation of the Min-property is verified for any $x \preceq y$ (where $\preceq$ is the ordering used in the transition monoid). In particular, the equation is true for the relation $\mathcal{R}$, which is a subset of any stabilisation order.

$4 \to 5$ uses the fact that Min-property is equational, so is preserved by quotients.

$5 \to 1$ is obtained by performing substitutions in computation trees: any idempotent node that is the descendant of a stabilisation node can be transformed into a stabilisation node itself. We get a normal form for computation trees over monoids of this fragment, that we call frontier trees. We then design min-automata that witness the existence of frontier trees for any input word.

We show $1 \to 6$ by adapting the classical automaton-to-expression algorithm performing inductive state removal. Here, new transitions are labeled by regular expressions together with an action from $\{\varepsilon, \mathtt{i}\}$. When the classical algorithm produces a Kleene star, we do so if the looping transition is labeled ε; otherwise if it is $\mathtt{i}$, we replace the Kleene star by $\leq n$, and the resulting edge is again labeled $\mathtt{i}$. This way, a Kleene star cannot be produced on top of a subexpression containing a $\leq n$.

We show $6 \to 7$ by induction on the structure of the expression. We build an ad-hoc generalisation of the expression$\to$ automaton algorithm, and show that the output B-automaton contains no reset.

Finally, $7 \to 1$ is obtained by observing that in the absence of resets, increments performed on k distinct counters can be performed on a single counter, by increasing the result at most k times. ◀

By a saturation algorithm, we can verify whether a given stabilisation monoid has Min-property in polynomial time. Hence by Theorem 2.15 and Theorem 2.16 we obtain:

▶ **Theorem 4.4.** *It is decidable if a regular cost function satisfies a condition of Theorem 4.2.*

5 Conclusion

We studied two dual classes of cost functions, defined by min-automata (also called distance automata), and max-automata. Both these classes have been studied in detail in other works. We showed that these classes enjoy many equivalent characterisations – such as restrictions of automata, logics, and expressions, and algebraic conditions. In both cases, the algebraic characterisation leads to decidability of membership in the class, in the spirit

of Schützenberger's seminal work on star-free languages [16]. Combining with the finite-index characterisation of regular cost functions from [17], we obtain a purely algebraic characterisation of cost functions defined by min- or max-automata.

References

1 Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In *CONCUR 2008*, volume 5201, pages 67–81, 2008.

2 Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *LICS 2013*, pages 13–22, 2013.

3 Mikolaj Bojanczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.

4 Mikolaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS 06*, pages 285–296, 2006.

5 Mikolaj Bojanczyk and Szymon Torunczyk. Deterministic automata and extensions of weak mso. In *FSTTCS*, pages 73–84, 2009.

6 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Automata, Languages and Programming, Internatilonal Collogquium, ICALP 2009, Proceedings, Part II*, pages 139–150, 2009.

7 Thomas Colcombet. *Fonctions régulières de coût*. Habilitation thesis, Université Paris Diderot–Paris, 2013.

8 Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013.

9 Thomas Colcombet and Laure Daviaud. Approximate comparison of distance automata. In *STACS 2013*, volume 20 of *LIPIcs*, pages 574–585, 2013.

10 Thomas Colcombet, Laure Daviaud, and Florian Zuleger. Size-change abstraction and max-plus automata. In *MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 208–219, 2014.

11 Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. Regular temporal cost functions. *Automata, Languages and Programming*, pages 563–574, 2010.

12 Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.

13 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3):405–425, 1994.

14 Denis Kuperberg. Linear temporal logic for regular cost functions. *Logical Methods in Computer Science*, 10(1), 2014.

15 Martin Lang, Christof Löding, and Amaldev Manuel. Definability and transformations for cost logics and automatic structures. In *MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2014.

16 M.P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190 – 194, 1965.

17 Szymon Torunczyk. *Languages of profinite words and the limitedness problem*. PhD thesis, PhD thesis, University of Warsaw, 2011.